A COMPARATIVE STUDY OF THE EFFICIENCY OF JET SCHEMES

PRINCE CHIDYAGWAI, JEAN-CHRISTOPHE NAVE, RODOLFO RUBEN ROSALES, AND BENJAMIN SEIBOLD

Abstract. We present two versions of third order accurate jet schemes, which achieve high order accuracy by tracking derivative information of the solution along characteristic curves. For a benchmark linear advection problem, the efficiency of jet schemes is compared with WENO and Discontinuous Galerkin methods of the same order. Moreover, the performance of various schemes in tracking solution contours is investigated. It is demonstrated that jet schemes possess the simplicity and speed of WENO schemes, while showing several of the advantages as well as the accuracy of DG methods.

1. Introduction

The advection of field quantities under a velocity field is an important sub-problem in many computational projects. Examples are the passive transport of concentrations, or the movement of interfaces using level set approaches [13]. We consider the linear advection equation

$$\phi_t + \vec{v} \cdot \nabla \phi = 0 \,, \tag{1}$$

which moves a scalar quantity $\phi(\vec{x}, t)$ by a given velocity field $\vec{v}(\vec{x}, t)$. Equation (1) is augmented with initial conditions $\phi(\vec{x}, 0) = \Phi(\vec{x})$ and boundary conditions. All data is assumed smooth in space and time. Furthermore, it is assumed that the problem at hand can be described by, or embedded into, a rectangular computational domain, equipped with a regular grid.

High order accurate numerical approximations of (1) on a fixed grid are commonly based on schemes that employ polynomials of sufficient degree to interpolate smooth solutions with the required accuracy. One popular type of approach are finite difference ENO [16] or WENO [11] methods. These store approximations of the solution values at the grid points, and achieve a high order polynomial approximation by considering local neighborhoods that are several grid points wide. Another type of approach are discontinuous Galerkin (DG) [14, 4, 5] methods. These achieve high order accuracy by storing a high degree polynomial approximation in each grid cell, and approximating the flux through cell boundaries based on a weak formulation of (1). Both WENO and DG methods are based on a semi-discretization of (1), and achieve high accuracy in time by using strong stability preserving (SSP) Runge-Kutta schemes [16, 6, 7].

Both types of approaches incur problems and difficulties. Due to the wide stencils of WENO methods, it is challenging to preserve high order near boundaries. Furthermore, their non-locality poses difficulties for an effective parallelization, and their use in conjunction with adaptive grid approaches [2]. In contrast, in DG methods, communication is limited to neighboring cells. However, DG approaches are characterized by costly quadratures over

2000 *Mathematics Subject Classification.* 65M25; 65M12; 35L04.

Key words and phrases. jet scheme, advection, high-order, WENO, DG, efficiency, contours.

grid cells and their edges, significant time step restrictions, and non-trivial implementation aspects.

A new class of approaches for (1), so called *jet schemes*, has been proposed recently [12, 15]. The goal of these methods is to provide an attractive compromise between WENO and DG methods, possessing the optimal locality and good resolution properties of DG, while being close to the computational efficiency and ease of implementation of WENO schemes. In this paper we conduct a comparative study of the efficiency and accuracy of this new class of methods. We consider two versions of third order accurate jet schemes, and compare them with WENO and DG methods of the same order. The considered jet schemes are described in Sect. 2. Then, in Sect. 3, DG methods are outlined, and in Sect. 4, information about the WENO schemes that we employ is given. Sect. 5 shows the numerical results obtained when applying the three types of approaches to two versions of a benchmark problem. In Sect. 6, a discussion of the observations and theoretical estimates of the computational cost are given.

2. Jet Schemes

Jet schemes are based on an advect–and–project approach in function spaces. Given an approximation to the solution of (1) at time t_n, an approximate solution at time $t_{n+1} = t_n + \Delta t$ is obtained by the time step

$$\phi^{n+1} = P \circ A_{t_{n+1},t_n} \phi^n \ .$$

Here A_{t_{n+1},t_n} is an approximate advection operator, defined by evolving the solution along characteristics using a numerical ODE solver, and P is a projection operator, given by a piecewise Hermite interpolation based on parts of the jet of the solution at the points of a cartesian grid. The fact that the projection requires data only at grid points allows to use this approach as a numerical scheme: when evaluating the advection operator, only the characteristic curves that go through grid points at t_{n+1} need to be considered. In addition, the evolution of derivatives of the solution along these characteristic must be found. As outlined below, the required spacial derivatives of the advection operator can be found by analytical differentiation (Sect. 2.3) or by approximations based on tracking multiple nearby characteristics (Sect. 2.5).

Jet schemes can be constructed in any space dimension, and for any order of accuracy [15]. For simplicity, here we only describe third order schemes in two space dimensions. We consider a rectangular computational domain $\Omega \subset \mathbb{R}^2$, equipped with a regular cartesian grid of grid size h.

2.1. **Projection.** In a grid cell $[a, a+h] \times [b, b+h] \subset \Omega$, let the vertices be indexed by a vector $\vec{q} \in \{0,1\}^2$, such that the vertex of index $\vec{q}$ is at position $\vec{x}_{\vec{q}} = (a + h\,q_1, b + h\,q_2)$. On each of the four vertices, let a vector of data be given by $\phi_{\vec{\alpha}}^{\vec{q}}\ \forall\, \vec{\alpha} \in \{0,1\}^2$. The data represents partial derivatives of orders up to 1 in each variable, as follows: $\vec{\alpha} = (0,0)$ represents function values ϕ; $\vec{\alpha} = (1,0)$ and $\vec{\alpha} = (0,1)$ represent first derivatives $\partial_x \phi$ and $\partial_y \phi$, respectively; and $\vec{\alpha} = (1,1)$ represents $\partial_{xy} \phi$. This data is interpolated by the bi-cubic polynomial

$$\mathcal{H}(\vec{x}) = \sum_{\vec{q},\vec{\alpha} \in \{0,1\}^p} \phi_{\vec{\alpha}}^{\vec{q}}\ W_{\vec{\alpha}}^{\vec{q}}(\vec{x}) \ , \tag{2}$$

where the $W_{\vec{\alpha}}^{\vec{q}}(\vec{x})$ are bi-cubic basis functions, given by the tensor product formulas

$$W_{\vec{\alpha}}^{\vec{q}}(\vec{x}) = h^{\alpha_1+\alpha_2}\ w_{\alpha_1}^{q_1}\left(\tfrac{x-a}{h}\right) w_{\alpha_2}^{q_2}\left(\tfrac{y-b}{h}\right) \ ,$$

and the w_α^q are the univariate basis functions

$$w_0^0(x) = 1 - 3x^2 + 2x^3, \quad w_0^1(x) = 3x^2 - 2x^3, \quad w_1^0(x) = x - 2x^2 + x^3, \quad w_1^1(x) = -x^2 + x^3.$$

The bi-cubic interpolant (2) is an $O(h^4)$ accurate approximation to any sufficiently smooth function ϕ that it interpolates on a cell of size h [15].

On the computational domain $\Omega \subset \mathbb{R}^2$, let the grid points be labeled $\vec{x}_{\vec{m}}$, where $\vec{m} \in \mathbb{Z}^2$. For any (sufficiently smooth) function $\phi : \Omega \to \mathbb{R}$, we define a global interpolant $\mathcal{H}_\phi$ as follows: at each grid point $\vec{x}_{\vec{m}}$, evaluate the derivatives of ϕ, $\partial_x\phi$, $\partial_y\phi$, and $\partial_{xy}\phi$, to produce a data vector $\phi_{\vec{\alpha}}^{\vec{m}}\ \forall\, \vec{\alpha} \in \{0,1\}^2$. Then, on each grid cell, use this data to define the bi-cubic interpolant (2). In the function space

$$S^{2,+} = \{\psi \in C^1 : \psi \text{ twice differentiable a.e. with } D^2\psi \in L^\infty\}\ ,$$

this procedure can be applied using the following convention: whenever $\partial_{xy}\psi$ must be evaluated at a point at which $D^2\psi$ is not defined in the classical sense, we define it as

$$\partial_{xy}\psi(\vec{x}_{\vec{m}}) = \tfrac{1}{2}\Big(\operatorname*{ess\,lim\,sup}_{\vec{x}\to\vec{x}_{\vec{m}}} \partial_{xy}\psi(\vec{x}) + \operatorname*{ess\,lim\,inf}_{\vec{x}\to\vec{x}_{\vec{m}}} \partial_{xy}\psi(\vec{x})\Big)\ .$$

The view on the general order of approximation case [15] reveals the rationale for the notation $S^{2,+}$. Clearly, the re-application of the interpolation procedure does not change the result: $\mathcal{H}_{\mathcal{H}_\phi} = \mathcal{H}_\phi$. Hence, in the space $S^{2,+}$ it can be formulated as a projection operator

$$P\phi = \mathcal{H}_\phi\ . \tag{3}$$

2.2. **Advection.** The characteristic form of equation (1) is

$$\frac{d\phi}{dt} = 0 \quad \text{along} \quad \frac{d\vec{x}}{dt} = \vec{v}(\vec{x},t)\ . \tag{4}$$

Let $\vec{X}(\vec{x},\tau,t)$ denote the solution of the ODE for the characteristic curves at time t, when starting with initial conditions $\vec{x}$ at time τ, i.e. it is defined by

$$\frac{\partial}{\partial t}\vec{X}(\vec{x},\tau,t) = \vec{v}(\vec{X}(\vec{x},\tau,t),t) \quad \text{with} \quad \vec{X}(\vec{x},\tau,\tau) = \vec{x}\ .$$

Then due to (4), the solution of (1) satisfies $\phi(\vec{x},\tau) = \phi(\vec{X}(\vec{x},\tau,t),t)$. In practice, the characteristic ODE (4) must be approximated. Let $\vec{\mathcal{X}}(\vec{x},\tau,t)$ represent an approximate solution of the ODE for the characteristic curves at time t, when starting with initial conditions $\vec{x}$ at time τ. It typically arises from a numerical ODE solver, e.g. a high order Runge-Kutta step. Furthermore, introduce the associated approximate advection operator $A_{\tau,t}$, which maps the solution at time t to an approximate solution at time τ. It acts on a function $g(\vec{x})$ as follows

$$(A_{\tau,t}\, g)(\vec{x}) = g(\vec{\mathcal{X}}(\vec{x},\tau,t))\ .$$

As shown in [12], the use of a locally k^{th} order Runge-Kutta scheme results in a k^{th} order accurate approximation to the solution

$$A_{t+\Delta t,t}\,\phi(\vec{x},t) - \phi(\vec{X}(\vec{x},t+\Delta t,t),t) = O(|\Delta t|^k)\ .$$

This means that through the characteristic equations (4), any ODE solver induces an approximate advection operator of the same order of accuracy. However, this idea alone cannot be used as a numerical time stepping scheme, since the step from t to $t+\Delta t$ in general generates a function $A_{t+\Delta t,t}\, g$ that cannot be represented with a finite amount of data. Therefore, at the end of every time step, we apply the projection operator (3), which generates a function that can be stored on a computer. The function space $S^{2,+}$ is invariant under diffeomorphisms,

thus both P and $A_{t+\Delta t,t}$ map from $S^{2,+}$ into itself. Consequently, one full approximate solution step is given by applying $P \circ A_{t+\Delta t,t}$ to the approximate solution at time t.

2.3. **Derivative Updates by Analytical Differentiation.** The application of the projection (3) requires the knowledge of ψ, $\partial_x\psi$, $\partial_y\psi$, $\partial_{xy}\psi$ at grid points, with $\psi = A_{t+\Delta t,t}\,\phi^n$. These spacial derivatives of $A_{t+\Delta t,t}\,\phi^n$ can be found by analytically differentiating the ODE solver, as demonstrated below for the Shu-Osher scheme [16]. One step with this scheme is $O((\Delta t)^4)$ accurate, which for the scaling $\Delta t \propto h$ matches the $O(h^4)$ accuracy of the projection operator (3). Using the notation $t_n = t$, $t_{n+1} = t + \Delta t$, and $t_{n+\frac{1}{2}} = t + \frac{1}{2}\Delta t$, we obtain the updates for the required parts of the jet ϕ, $\nabla\phi = (\partial_x\phi, \partial_y\phi)$, and $\partial_{xy}\phi$ at a grid point $\vec{x}$ as:

$$
\begin{aligned}
\vec{x}_1 &= \vec{x} - \Delta t\,\vec{v}(\vec{x}, t_{n+1}) \\
\nabla\vec{x}_1 &= I - \Delta t\,\nabla\vec{v}(\vec{x}, t_{n+1}) \\
\partial_{xy}\vec{x}_1 &= -\Delta t\,\partial_{xy}\vec{v}(\vec{x}, t_{n+1}) \\
\vec{x}_2 &= \tfrac{3}{4}\vec{x} + \tfrac{1}{4}\vec{x}_1 - \tfrac{1}{4}\Delta t\,\vec{v}(\vec{x}_1, t_n) \\
\nabla\vec{x}_2 &= \tfrac{3}{4}I + \tfrac{1}{4}\nabla\vec{x}_1 - \tfrac{1}{4}\Delta t\,\nabla\vec{x}_1 \cdot \nabla\vec{v}(\vec{x}_1, t_n) \\
\partial_{xy}\vec{x}_2 &= \tfrac{1}{4}\partial_{xy}\vec{x}_1 - \tfrac{1}{4}\Delta t(\partial_{xy}\vec{x}_1 \cdot \nabla\vec{v}(\vec{x}_1, t_n) + ((\partial_x\vec{x}_1)^T \cdot (\partial_y\vec{x}_1)) : D^2\,\vec{v}(\vec{x}_1, t_n)) \\
\vec{x}_{\text{foot}} &= \tfrac{1}{3}\vec{x} + \tfrac{2}{3}\vec{x}_2 - \tfrac{2}{3}\Delta t\,\vec{v}(\vec{x}_2, t_{n+\frac{1}{2}}) \\
\nabla\vec{x}_{\text{foot}} &= \tfrac{1}{3}I + \tfrac{2}{3}\nabla\vec{x}_2 - \tfrac{2}{3}\Delta t\,\nabla\vec{x}_2 \cdot \nabla\vec{v}(\vec{x}_2, t_{n+\frac{1}{2}}) \\
\partial_{xy}\vec{x}_{\text{foot}} &= \tfrac{2}{3}\partial_{xy}\vec{x}_2 - \tfrac{2}{3}\Delta t(\partial_{xy}\vec{x}_2 \cdot \nabla\vec{v}(\vec{x}_2, t_{n+\frac{1}{2}}) + ((\partial_x\vec{x}_2)^T \cdot (\partial_y\vec{x}_2)) : D^2\,\vec{v}(\vec{x}_2, t_{n+\frac{1}{2}})) \\
\phi(\vec{x}, t_{n+1}) &= \mathcal{H}(\vec{x}_{\text{foot}}, t_n) \\
(\nabla\phi)(\vec{x}, t_{n+1}) &= \nabla\vec{x}_{\text{foot}} \cdot \nabla\mathcal{H}(\vec{x}_{\text{foot}}, t_n) \\
(\partial_{xy}\phi)(\vec{x}, t_{n+1}) &= \partial_{xy}\vec{x}_{\text{foot}} \cdot \nabla\mathcal{H}(\vec{x}_{\text{foot}}, t) + ((\partial_x\vec{x}_{\text{foot}})^T \cdot (\partial_y\vec{x}_{\text{foot}})) : D^2\mathcal{H}(\vec{x}_{\text{foot}}, t_n)
\end{aligned}
\tag{5}
$$

In this approach, the characteristic curve is tracked from $\vec{x}$ at time $t + \Delta t$ back to $\vec{x}_{\text{foot}}$ at time t. The data at this position is given by the Hermite interpolation, defined by the data (at time t) at the four vertices of the cell that $\vec{x}_{\text{foot}}$ is contained in. The update rules for the derivatives are systematically inherited from the update rule for the function value, and they match exactly what one would obtain when evolving the solution using $A_{t+\Delta t,t}$ everywhere, and then applying the spacial derivatives.

2.4. **Order of Accuracy and Stability.** One step of the jet scheme described above is fourth order accurate, since the advection operator is $O((\Delta t)^4)$ accurate, and the projection operator is $O(h^4)$ accurate. As usual, when going from this local error to the global error, one order is lost, since $O(\frac{1}{\Delta t})$ time steps are required to reach the final time. Hence, the presented scheme is globally third order, given that it is stable.

Like in many other numerical approaches, stability is not automatically guaranteed. As shown in [15], jet schemes can be constructed (by using a different projection than (3)) that are unstable. However, the jet scheme based on the projection (3) is stable. A key factor in the stability argument is that among all (sufficiently smooth) functions that match given data on a cartesian grid, the Hermite interpolant (2) minimizes the stability functional

$$\mathcal{F}[\phi] = \int_\Omega (\partial_{xxyy}\phi(\vec{x}))^2\,\mathrm{d}\vec{x}\,.$$

Hence $\mathcal{F}[P\phi] \leq \mathcal{F}[\phi]$ for all sufficiently smooth ϕ, which yields bounds on the amounts of oscillations that the numerical scheme can create [15].

2.5. **A Simpler and More Efficient Derivative Tracking.** The analytical differentiation procedure, outlined with example (5), is the most general approach to derive the update rules for derivatives. However, it is not always the simplest to implement. An alternative approach, which is solely based on tracking function values, is provided by *ε-finite differences*. For the here considered third order jet scheme in 2D, the following procedure can be employed. As shall be seen in Sect. 5, it leads to a very simple and efficient numerical scheme.

At a grid point $\vec{x} = (x, y)$, instead of tracking one characteristic curve from time t_{n+1} back to time t_n, we track four characteristic curves, starting at $\vec{x}^{\vec{q}} = (x + q_1\varepsilon, y + q_2\varepsilon)$ where $\vec{q} \in \{-1, 1\}^2$, and ε is a small number, see below. Let the corresponding characteristic foot-points be denoted $\vec{x}^{\vec{q}}_{\text{foot}}$. The updates for the required derivatives are obtained as follows:

(1) The "center of mass" $\frac{1}{4}\big(\vec{x}^{(1,1)}_{\text{foot}} + \vec{x}^{(-1,1)}_{\text{foot}} + \vec{x}^{(1,-1)}_{\text{foot}} + \vec{x}^{(-1,-1)}_{\text{foot}}\big)$ determines to which cell all four foot-points are associated to.
(2) The Hermite interpolant (2) that corresponds to that cell is evaluated at each foot-point to yield the values $\phi(\vec{x}^{\vec{q}}, t_{n+1}) = \mathcal{H}(\vec{x}^{\vec{q}}_{\text{foot}}, t_n)$.
(3) The required parts of the jet are defined as

$$\begin{aligned}
\phi(\vec{x}, t_{n+1}) &= \tfrac{1}{4}\left(\phi(\vec{x}^{(1,1)}, t_{n+1}) + \phi(\vec{x}^{(-1,1)}, t_{n+1}) + \phi(\vec{x}^{(1,-1)}, t_{n+1}) + \phi(\vec{x}^{(-1,-1)}, t_{n+1})\right)\\
\partial_x\phi(\vec{x}, t_{n+1}) &= \tfrac{1}{4\varepsilon}\left(\phi(\vec{x}^{(1,1)}, t_{n+1}) - \phi(\vec{x}^{(-1,1)}, t_{n+1}) + \phi(\vec{x}^{(1,-1)}, t_{n+1}) - \phi(\vec{x}^{(-1,-1)}, t_{n+1})\right)\\
\partial_y\phi(\vec{x}, t_{n+1}) &= \tfrac{1}{4\varepsilon}\left(\phi(\vec{x}^{(1,1)}, t_{n+1}) + \phi(\vec{x}^{(-1,1)}, t_{n+1}) - \phi(\vec{x}^{(1,-1)}, t_{n+1}) - \phi(\vec{x}^{(-1,-1)}, t_{n+1})\right)\\
\partial_{xy}\phi(\vec{x}, t_{n+1}) &= \tfrac{1}{4\varepsilon^2}\left(\phi(\vec{x}^{(1,1)}, t_{n+1}) - \phi(\vec{x}^{(-1,1)}, t_{n+1}) - \phi(\vec{x}^{(1,-1)}, t_{n+1}) + \phi(\vec{x}^{(-1,-1)}, t_{n+1})\right)
\end{aligned}$$

All approximations are $O(\varepsilon^2)$ accurate. In addition, round-off errors of up to $O(\delta/\varepsilon^2)$ arise, where δ is the accuracy of the floating point operations. With the optimal choice $\varepsilon = O(\delta^{1/4})$, the errors incurred by the ε-finite differences are of magnitude $O(\delta^{1/2})$. Thus, with double precision arithmetics, the presented approach can be used up to a desired accuracy of 10^{-7}.

3. Discontinuous Galerkin Method

DG methods [14, 4] have a wide area of application. They can successfully approximate nonlinear problems on unstructured geometries, and thus are much more general than the problems considered in this paper. However, since the task at hand is the numerical approximation of the linear advection equation (1) on a regular grid, it is natural to investigate the accuracy and efficiency of DG methods for it. The DG methodology that we use here is in line with the ideas presented in [3, 5]. For convenience, we restrict the presentation here to incompressible velocity fields $\nabla \cdot \vec{v} = 0$. In this case, the advection equation (1) can be written in conservative form as

$$\phi_t + \nabla \cdot (\vec{v}\phi) = 0\,. \tag{6}$$

Equation (6) is discretized in space as follows. Let $\mathcal{T}_h$ be a regular triangulation of the computational domain Ω. At each instance in time, we seek an approximation ϕ_h of ϕ, such that $\phi_h(t)$ belongs to the finite dimensional space

$$S_h^k = \{\psi \in L^1(\Omega) : \psi|_K \in P_h^k(K)\ \forall\, K \in \mathcal{T}_h\}\,,$$

where $P_h^k(K)$ denotes the space of polynomials of degree $\leq k$ that live on the element K. Here, we choose $k = 2$ to obtain a third order scheme. As in the standard DG formulation [3], we replace ϕ by ϕ_h in (6), multiply by a test function $\psi_h \in S_h^k(K)$, and integrate over $K \in \mathcal{T}_h$ to obtain the semi-discrete formulation

$$\frac{d}{dt}\int_K \phi_h \psi_h \,\mathrm{d}x = \int_K \vec{v}\cdot\nabla\psi_h \phi_h \,\mathrm{d}x - \int_{e\in\partial K} \widehat{\phi_h}\,\vec{v}\cdot\vec{n}_{e,K}\psi_h \,\mathrm{d}\Gamma \quad \forall\, \psi_h \in S_h^k(K)\,. \tag{7}$$

Here $\vec{n}_{e,K}$ is the outward unit normal on edge e, and $\widehat{\phi_h}$ is an expression that comprises the values of ϕ_h on each side of the edge. We use an upwind flux, which is defined as

$$\widehat{\phi_h} = \begin{cases} \phi_h^+ & \text{if } \vec{v}\cdot\vec{n}_{e,K} \geq 0 \\ \phi_h^- & \text{otherwise}\,, \end{cases}$$

where $\phi_h^\pm = \lim_{\varepsilon\to 0^\pm} \phi_h(\vec{x} + \varepsilon\vec{n}_{e,K}, t)$, and the velocity field is evaluated in the edge center. The integrals over edges and elements in (7) are approximated by Gaussian quadrature rules that are exact for polynomials of degree $2k$ over the elements, and exact over polynomials of degree $2k+1$ over the edges. This results in an ODE system of the form

$$\frac{d}{dt}\phi_h = L_h(\phi_h) \quad \text{with} \quad \phi_h(\vec{x},0) = \Phi_h(\vec{x})\,, \tag{8}$$

where $L_h(\phi_h)$ is the spacial discretization of the operator $-\nabla\cdot(\vec{v}\phi)$, and $\Phi_h(\vec{x}) \in S_h^k$ approximates $\Phi(\vec{x})$. The time integration of (8) is done using the third order SSP Shu-Osher scheme [16]. Stability is ensured if the specific CFL condition

$$\Delta t < \tfrac{1}{c(2k+1)}\, h$$

with $c > 1$ is guaranteed [5]. For the examples considered in this paper, the choice $c = 2$ turns out to yield the lowest cost vs. accuracy ratio. Since for the numerical tests conducted here the solutions are very smooth, no slope limiters are implemented.

The DG code used here is based on triangular elements. In order to have a fair comparison with the cartesian grids that WENO and jet schemes use, we design meshes as follows. For a desired resolution h, first a cartesian grid of resolution $\sqrt{2}h$ is constructed. Then each cell is divided along its diagonal into two triangles. Like a cartesian grid of resolution h, the resulting triangular mesh consists of $1/h^2$ elements, and the shortest height in an element is h.

4. WENO Scheme

Like DG methods, WENO schemes [11] are based on a semi-discretization of (1) in space, and the use of SSP schemes [16, 6] to advance in time. Unlike DG methods, WENO schemes are specific to regular grids. We employ the third order WENO finite difference approach described in [9], using the Shu-Osher scheme [16] with $\Delta t = h$ in time. For the spacial approximation, equation (1) is rewritten as

$$\phi_t = -u\,\partial_x\phi - v\,\partial_y\phi\,.$$

At each grid point, both derivatives $\partial_x\phi$ and $\partial_y\phi$ are approximated in a univariate fashion by using values on four grid points in each coordinate direction: two in the upwind direction, and one in the downwind direction. WENO schemes have limiters built into the derivative

approximations: smoothness indicators yield a nonlinear weighted average of different polynomial approximations of the derivatives. We consider two versions of limiting: one as described in [9], and another one without limiting (which yields a linear finite difference scheme).

5. Numerical Results

We test the accuracy, the relative efficiency, and the quality of tracking contours of jet schemes, DG methods, and WENO approaches using the classical *vortex in a box* flow test [1, 10], adapted as follows. On the computational domain $(x, y) \in [0, 1]^2$, and for $t \in [0, T]$, we consider the linear advection equation (1) with the velocity field

$$\vec{v}(x, y, t) = \cos\left(\pi \tfrac{t}{T}\right) \begin{pmatrix} \sin^2(\pi x)\, \sin(2\pi y) \\ -\sin(2\pi x)\, \sin^2(\pi y) \end{pmatrix}, \tag{9}$$

which is a model for the passive swirling and successive un-swirling of a concentration field by an incompressible fluid motion. This test is a mathematical analog of well-known "unmixing" experiments [8, 17]. For this flow, the final solution equals the initial conditions, i.e. $\phi(x, y, T) = \phi(x, y, 0)$. We consider smooth initial conditions, and periodic boundary conditions. Note that for linear advection problems, this test is quite general. Moreover, it has a built-in difficulty parameter: the larger T, the more challenging it is to numerically resolve the highly elongated contours of the solution.

5.1. **Numerical Efficiency.** For the investigation of numerical efficiency, we consider smooth periodic initial conditions $\phi(x, y, 0) = \cos(2\pi x) \cos(4\pi y)$, and choose $T = 1$. For various mesh resolutions h, we apply five different numerical schemes to the test problem. The results are shown in Figs. 1–3. Specifically, we consider:

(i) A classical third order WENO scheme (see Sect. 4), denoted by large squares;
(ii) A third order WENO scheme without limiting (see Sect. 4), denoted by small squares;
(iii) A third order DG method (see Sect. 3), denoted by triangles;
(iv) A third order jet scheme with a full derivative update (see Sect. 2.3), denoted by circles;
(v) A third order jet scheme based on ε-finite differences (see Sect. 2.5), denoted by stars.

For all methods, we measure the error at $t = T$ in the L^∞ norm, and the CPU time. Figure 1 shows the error as a function of the resolution h. One can observe that jet schemes and DG have roughly the same accuracy. In contrast, for the same resolution, WENO schemes yield significantly larger errors. It is also visible that the WENO3 with limiters does not achieve the full third order accuracy on the range of resolutions under consideration (see [9]). It is apparent that for the smooth solution at hand, limiters are not beneficial. Figure 2 shows the CPU time as a function of the resolution h. One can see a clear ranking in computational costs: WENO schemes are very fast, jet schemes are more costly, and DG is even more costly. It is visible that ε-finite differences are not only simpler to implement, but are also significantly faster than the jet scheme based on the full derivative update. Finally, Fig. 3 shows the CPU time as a function of the error. This cost vs. accuracy ratio measures the true efficiency of the numerical schemes. The results show that the low computational cost of WENO schemes renders them preferable over DG. However, WENO does not possess the optimal locality that DG has. Jet schemes provide an interesting alternative. While the full update version is slightly less efficient than the non-limited WENO, the ε-finite differences version is even more efficient.

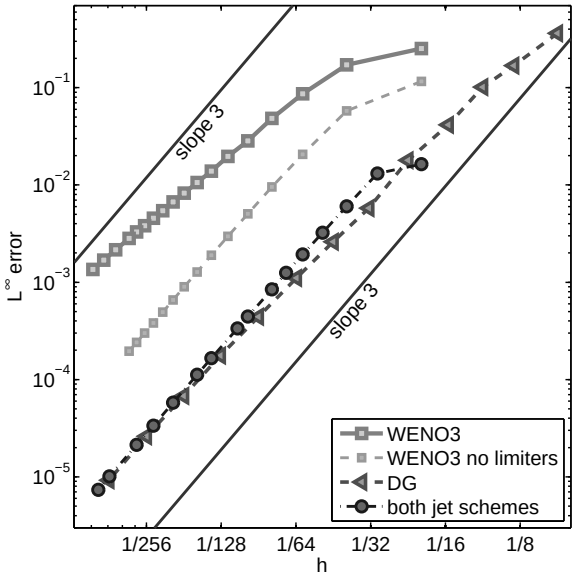

Figure 1. Error convergence for third order jet schemes, DG, and WENO.

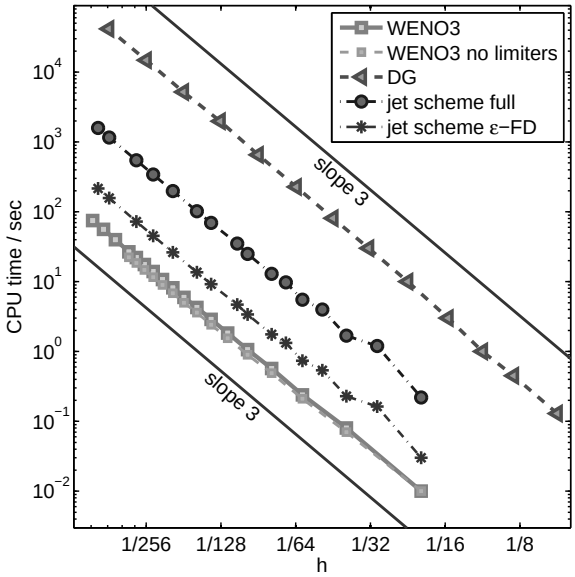

Figure 2. Scaling of the computational cost for jet schemes, DG, and WENO.

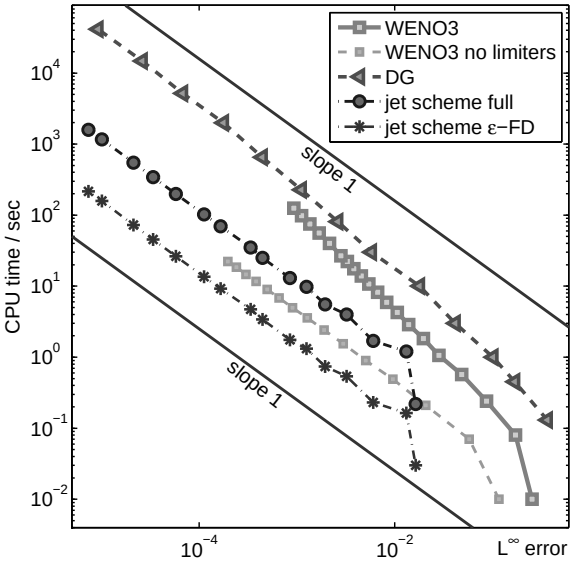

Figure 3. Efficiency of jet schemes, DG, and WENO, measured as cost over accuracy.

5.2. **Quality of Contours.** In order to assess the quality of the numerical approximations that the different methods produce, we consider how accurately contours of the solution are deformed under the flow (9) with $T = 6$. For this test, we choose the initial conditions

$$\phi(x, y, 0) = \exp\left(-10(x - 0.5)^2 - 10(x - 0.75)^2\right) ,$$

and consider three contours at $\phi = \exp(-10\, r_k^2)$, where $r_1 = 0.044$, $r_2 = 0.132$, and $r_3 = 0.220$. At $t = 0$, these contours are three concentric circles of radii r_k, centered at the point $(0.5, 0.75)$. We plot the deformed contours at $t = \frac{T}{2} = 3$, the time of their maximum deformation.

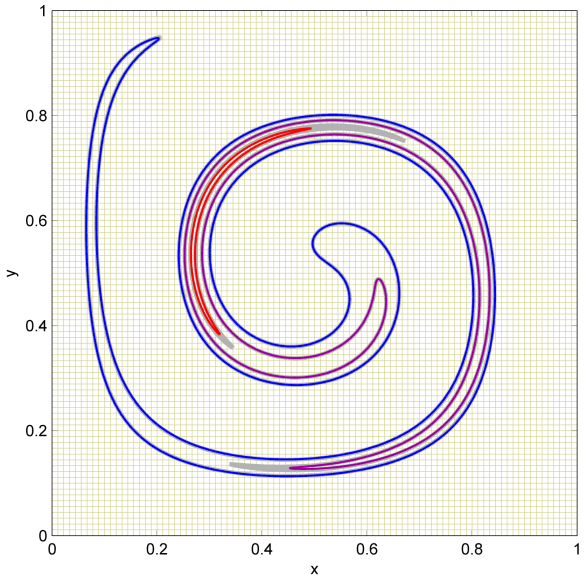

Figure 4. Deformed contours, computed with a jet scheme using $h = 1/90$.

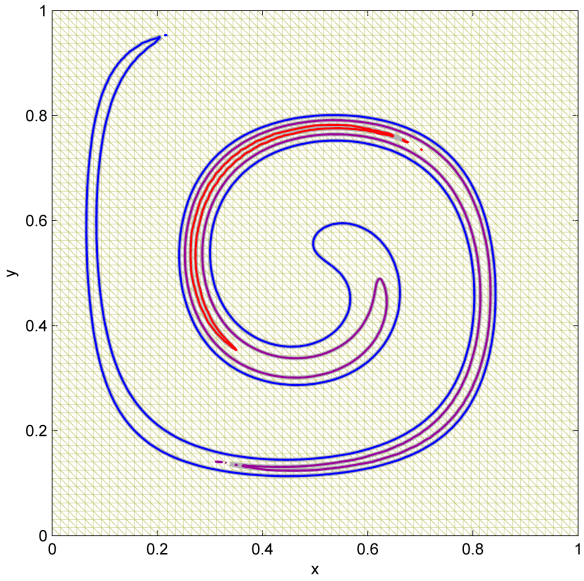

Figure 5. Deformed contours, computed with a DG scheme using $h = 1/90$.

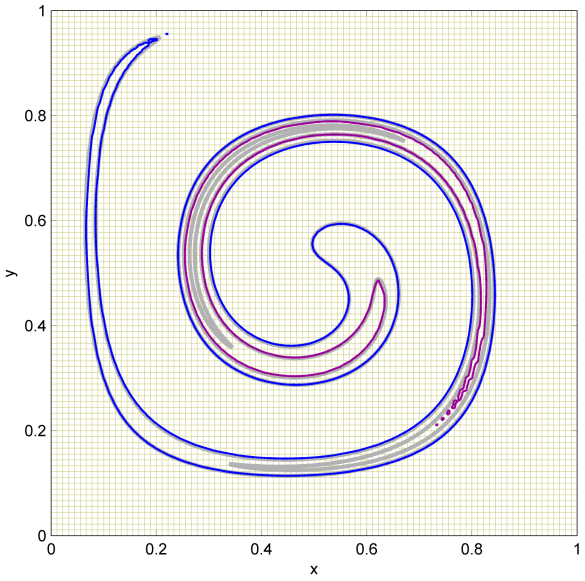

Figure 6. Deformed contours, computed with a WENO scheme using $h = 1/90$.

We compare a jet scheme, DG, and WENO for the same resolution h. As found in Sect. 5.1, these methods involve very different computational costs. Yet, this equal-h comparison is of practical relevance, since frequently the advection equation (1) is only a sub-problem in a larger project, and the computational grid is determined by the underlying application. Fig. 4 shows the contours obtained with a jet scheme based on ε-finite differences (see Sect. 2.5); Fig. 5 shows the contours obtained with DG (see Sect. 3); and Fig. 6 shows the contours obtained with WENO without limiting (see Sect. 4). In all cases, the inner (red) contour is $\phi = \exp(-10\, r_1^2)$, the middle (purple) contour is $\phi = \exp(-10\, r_2^2)$, and the outer (blue) contour

is $\phi = \exp(-10\, r_3^2)$. Moreover, each contour underlays the contour of the true solution as a thick gray curve. In addition, the computational grid is shown in the background. Note that in all cases, contours are plotted using a sub-grid: for jet schemes, the solution inside a grid cell is given by the Hermite bi-cubic interpolant (2); for DG, the solution inside an element is given by a bivariate quadratic polynomial; and for WENO, we use a simple bi-linear interpolant on each grid cell.

As it was already visible in the error convergence graph shown in Fig. 1, also in terms of the tracking of contours, the jet scheme as well as DG are significantly more accurate than WENO. The comparison between jet schemes and DG is more interesting. Clearly, the DG contours shown in Fig. 5 are closer to the true solution's contours than the jet scheme's contours are. However, at the tips of contours, DG produces small ripples and break-ups. In contrast, the jet scheme contours are extremely uniform, all the way to their end. Moreover, for the tracking of contours, DG has another potential disadvantage: due to the discontinuities of the numerical solution across element boundaries, the resulting contours may have small jumps (in fact, the contours in Fig. 5 possess jumps, but they are too small to be visible to the eye). In certain applications, such as level set methods [13], this could lead to problems. These observations come in addition to the fact that for the same resolution, DG is significantly more costly than jet schemes (see Fig. 2).

6. Discussion

The two types of traditional numerical methods, WENO and DG, achieve high order accuracy in very different ways: WENO considers function values in a wide neighborhood. In contrast, DG uses high order polynomials in each element. This gives DG a certain level of sub-grid resolution, and restricts communication to neighboring elements only. While based on a very different methodology, the recently proposed [12, 15] jet schemes share these advantages with DG.

The results presented here show that the advantages of DG come at the expense of efficiency. This observation can be explained by the following theoretical estimates. With a third order Runge-Kutta scheme, all presented schemes require three right hand side evaluations per time step. On a mesh of N elements, even a fully optimized and problem-specific DG code requires at least $11.5N$ evaluations of the velocity field per right hand side evaluation (7 quadrature points on each of N elements, and 3 quadrature points on each of $\frac{3}{2}N$ edges). In comparison, WENO required N velocity field evaluations, and a jet scheme (using ε-finite differences) requires $4N$ velocity field evaluations (4 characteristics per grid point). In addition, due to its more restrictive CFL condition, DG requires 5–10 times more time steps than WENO or jet schemes. The results shown in Fig. 2 are in line with these theoretical estimates, albeit at more extreme ratios between the costs of the methods. These increased ratios are most likely due to additional data structure management requirements for DG, and also for jet schemes. However, the key results in the efficiency plot shown in Fig. 3 remain valid even if the measured CPU times were replaced by the theoretical cost estimates given above.

In terms of accuracy, jet schemes and DG yield similar accuracies (for the same number of elements). In contrast, the low cost of WENO is outweighed by its lower accuracy. Therefore, in terms of true efficiency, jet schemes and WENO are in the same range, while DG methods are significantly more costly. This observation is further confirmed by a comparison of the quality of contour approximation.

These investigations lead to the conclusion that jet schemes indeed fill a need as computational approaches for advection problems on regular grids, namely: they are simpler to implement, and less costly than DG. However, in contrast to WENO, they are optimally local and thus are preferable for the purpose of parallelization, adaptive mesh refinement, and the implementation of complicated boundary conditions. In addition, jet schemes impose no restrictions on the ODE solvers that are needed. In contrast, DG and WENO methods require SSP ODE solvers to ensure TVD stability [7]. The key weakness of jet schemes is their limited range of applicability. While established for linear advection [12, 15], their application to more general advection problems is a the topic of current research.

Acknowledgments

The authors would like to acknowledge the support by the National Science Foundation. J.-C. Nave, R. R. Rosales, and B. Seibold were supported through grant DMS–0813648, and P. Chidyagwai was supported through grant DMS–1115269. In addition, P. Chidyagwai, R. R. Rosales, and B. Seibold wish to acknowledge partial support by the National Science Foundation through grants DMS–1007967, DMS–1007899, DMS–1115269, and DMS–1115278. Further, J.-C. Nave wishes to acknowledge partial support by the NSERC Discovery Program. This research was supported in part by the National Science Foundation through major research instrumentation grant number CNS–09–58854.

References

[1] J. B. Bell, P. Colella, and H. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85(2):257–283, 1989.

[2] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53(3):484–512, 1984.

[3] B. Cockburn. An introduction to the Discontinuous Galerkin method for convection-dominated problems. C.I.M.E. Lecture notes, 1997.

[4] B. Cockburn and C.-W. Shu. The local Discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463, 1988.

[5] B. Cockburn and C.-W. Shu. Runge-Kutta Discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3):173–261, 2001.

[6] S. Gottlieb and C.-W. Shu. Total variation diminishing Runge-Kutta schemes. *Math. Comp.*, 67(221):73–85, 1998.

[7] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability preserving high order time discretization methods. *SIAM Rev.*, 43(1):89–112, 2001.

[8] J. P. Heller. An unmixing demonstration. *Am. J. Phys.*, 28:348–353, 1960.

[9] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126(1):202–228, 1996.

[10] R. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, 33(2):627–665, 1996.

[11] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 115(1):200–212, 1994.

[12] J.-C. Nave, R. R. Rosales, and B. Seibold. A gradient-augmented level set method with an optimally local, coherent advection scheme. *J. Comput. Phys.*, 229(10):3802–3827, 2010.

[13] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[14] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[15] B. Seibold, R. R. Rosales, and J.-C. Nave. Jet schemes for advection problems. *Discrete Contin. Dyn. Syst. Ser. B*, 17(4):1229–1259, 2012.

[16] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.
[17] G. I. Taylor. Low reynolds number flow. Movie, 1961. U.S. National Committee for Fluid Mechanics Films (NCFMF).

(Prince Chidyagwai) Department of Mathematics, Temple University,
1805 North Broad Street, Philadelphia, PA 19122

E-mail address: chidyagp@temple.edu

URL: http://www.math.temple.edu/~chidyagp

(Jean-Christophe Nave) Department of Mathematics and Statistics, McGill University,
805 Sherbrooke W., Montreal, QC, H3A 2K6, Canada

E-mail address: jcnave@math.mcgill.ca

URL: http://www.math.mcgill.ca/jcnave

(Rodolfo Ruben Rosales) Department of Mathematics, Massachusetts Institute of Technology,
77 Massachusetts Avenue, Cambridge, MA 02139

E-mail address: rrr@math.mit.edu

(Benjamin Seibold) Department of Mathematics, Temple University,
1805 North Broad Street, Philadelphia, PA 19122

E-mail address: seibold@temple.edu

URL: http://www.math.temple.edu/~seibold